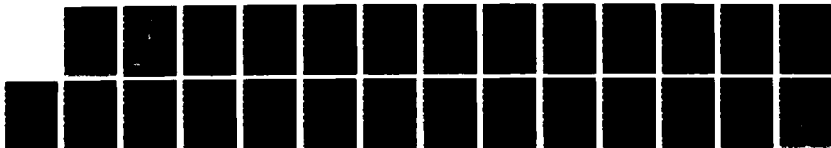
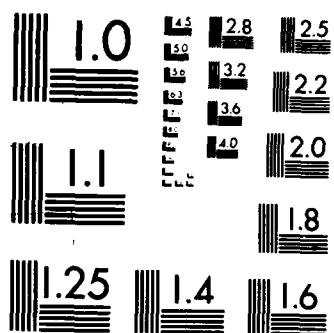


AD-A192 762 AN O(N²) METHOD FOR COMPUTING THE EIGENSYSTEM OF N X N 1/1
SYMMETRIC TRIIDIAGO (U) INSTITUTE FOR COMPUTER
APPLICATIONS IN SCIENCE AND ENGINEERIN D GILL ET AL
UNCLASSIFIED MAR 88 ICASE-88-19 NASA-CR-181647 F/G 12/1 NL





DTIC FILE COPY

NASA Contractor Report 181647

ICASE REPORT NO. 88-19

2

ICASE

AN $O(N^2)$ METHOD FOR COMPUTING THE EIGENSYSTEM OF $N \times N$
SYMMETRIC TRIDIAGONAL MATRICES BY THE DIVIDE AND CONQUER
APPROACH

AD-A192 762

Doron Gill
Eitan Tadmor

DTIC
ELECTE
APR 08 1988
S & D

Contract No. NAS1-18107
March 1988

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center, Hampton, Virginia 23665

Operated by the Universities Space Research Association

NASA

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

88 4 8 041

An $O(N^2)$ Method for Computing the Eigensystem of $N \times N$ Symmetric Tridiagonal Matrices by the Divide and Conquer Approach

Doron Gill and Eitan Tadmor*
School of Mathematical Sciences
Raymond and Beverly Sackler Faculty of Exact Sciences
Tel Aviv University
Tel Aviv, Israel
and

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia

Abstract

We propose an efficient method to solve the eigenproblem of $N \times N$ Symmetric Tridiagonal (ST) matrices. Unlike the standard eigensolvers which necessitate $O(N^3)$ operations to compute the eigenvectors of such ST matrices, the proposed method computes both the eigenvalues and eigenvectors with only $O(N^2)$ operations. The method is based on serial implementation of the recently introduced Divide and Conquer (DC) algorithm [3],[1],[4]. It exploits the fact that by $O(N^2)$ of DC operations, one can compute the eigenvalues of $N \times N$ ST matrix and a finite number of pairs of successive rows of its eigenvector matrix. The rest of the eigenvectors—all of them or one at the time, are computed by linear three-term recurrence relations. We conclude with numerical examples, which demonstrate the superiority of the proposed method by saving an order of magnitude in execution time at the expense of sacrificing a few orders of accuracy.

Availability Codes	
Dist	Avail and/or Special
A-1	INSPECTED 4

This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-18107 while the second author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23665. Additional support was provided by U.S.-Israel BSF Grant No. 85-00346, and by NSF Grant No. DMS85-03294 and ARO Grant No. DAAG-85-K-0190 while in residence at UCLA, Los Angeles, CA 90024.

*Bat-Sheva Foundation Fellow.

1. INTRODUCTION

The QR algorithm computes the eigenvalues of an $N \times N$ Symmetric Tridiagonal (ST) matrix with $O(N^2)$ operations, while the corresponding eigenvector matrix is accumulated during the algorithm at the expense of $O(N^3)$ operations. The additional order of magnitude required to compute the eigenvectors is typical of serial algorithms. A complete $O(N^2)$ eigensolver can be obtained by appending such serial algorithms with the Inverse Iteration (INVIT) method. Indeed, $O(N)$ operations of only one INVIT will suffice to accurately compute each eigenvector corresponding to an isolated eigenvalue [8, Chapter 4]. In case of clustered eigenvalues, however, the INVIT requires a more carefully chosen initialization, in order to avoid the loss of mutual orthogonality between the corresponding, closely "related", eigenvectors.¹

Recently, a parallel Divide and Conquer algorithm was introduced for computing the spectral decomposition of ST matrices [3],[1],[4]. A *serial* implementation of this algorithm, described in Section 2, requires the same number of operations. Namely, the eigenvalues – which coincide with the roots of the so-called secular equation [6], are computed at the expense of no more than $O(N^2)$ sequential operations, while the associated eigenvectors necessitate $O(N^3)$ sequential operations. As before, the INVIT – taken with the necessary precautions, is available here as an $O(N^2)$ method to compute these eigenvectors. In Sections 3 and 4 we propose an alternative efficient method, derived from (and therefore better suits) the DC algorithm, which computes the eigensystem of $N \times N$ ST matrices with only $O(N^2)$ sequential operations. The method employs linear three term recurrence relations which successively compute the *rows* of the eigenvector matrix (or the *components* of each of the desired eigenvectors). The coefficients of these relations depend on the already computed eigenvalues, and the method hinges on the fact that the initial first two rows (or components) for the recurrence relations emerge naturally from the DC computation of these eigenvalues. Thus, the input data for the recurrence relations depend

¹We thank Professor Beresford Parlett for an enlightening discussion on this issue.

solely on the $O(N^2)$ operations for the *DC* calculation of the eigenvalues. Together with the additional $O(N^2)$ operations required to carry out these relations, we end up with an efficient $O(N^2)$ method to compute the whole eigensystem of *ST* matrices. It should be emphasized that the advantages of the *DC* algorithm are retained in our case. That is, we have a method which on the one hand is well suited to exploit parallelism, while on the other hand, even when run in serial mode on large problems, the method is faster than the previously best sequential algorithms, e.g , [3],[4].

Due to the sensitivity of the three term recurrence relations, their input data should be provided with high accuracy. To achieve this, we employ in Section 5 an improved root finder – interesting for its own sake – in order to solve the secular equation mentioned above. Numerical examples which demonstrate the efficiency as well as the limitations of the proposed method are presented in Section 6.

2. THE DIVIDE AND CONQUER ALGORITHM - AN OVERVIEW

Let D_N be an $N \times N$ diagonal matrix and let $D_N + \sigma z_N z_N^t$ be a Rank One Modification (ROM) of this matrix by a unit N -vector z_N .² The spectral decomposition of such ROM matrices is in the heart of the Divide and Conquer (DC) algorithm. Here we note that the problem of finding the spectral decomposition of an N -dimensional ROM matrix, the so-called *updating problem*, can be solved at the expense of no more than $\text{Const.} N^2$ operations [1],[3],[4]. Details of this solution are discussed in Section 4.

With this in mind we now turn to consider the eigenproblem of general $N \times N$ Symmetric Tridiagonal (ST) matrices

$$T_N = \begin{bmatrix} t_{11} & t_{12} & & & \\ & t_{21} & t_{22} & & \\ & & & \ddots & \\ & & & & t_{mm} & \cdot & t_{mm+1} \\ \dots & & & & & & \dots & \\ & & & & t_{m+1,m} & \cdot & t_{m+1,m+1} & \\ & & & & & \ddots & & \\ & & & & & & & t_{N-1,N} \\ & & & & & & & t_{N,N-1} & t_{NN} \end{bmatrix}, \quad t_{ij} = t_{ji}.$$

We can assume without restriction that $N = 2m$ is even, and that T_N is already given in its unreduced form, i.e., $t_{i,i+1} \neq 0$, $1 \leq i \leq N-1$, for otherwise T_N is decoupled into smaller unreduced ST matrices. Then, we can *split* T_N into the sum of

$$T_N = \begin{bmatrix} t_{11} & t_{12} & & & \\ & t_{22} & & & \\ & & t_{mm} - \beta & \cdot & 0 \\ \dots & & & & \dots & & 0 \\ & & & & 0 & \cdot & t_{m+1,m+1} - \beta \\ & & & & & \ddots & \\ & & & & & & t_{N,N} \end{bmatrix} + \beta \begin{bmatrix} & & & & \\ & 0 & \cdot & 0 & \\ & & 1 & 1 & \\ \dots & & 1 & 1 & \dots \\ & & & 1 & 1 \\ & 0 & \cdot & 0 & \\ & & & & \cdot \end{bmatrix}, \quad \beta = t_{m,m+1},$$

i.e.,

$$T_N = \begin{bmatrix} T_{\frac{N}{2}}^{(1)} & \\ & T_{\frac{N}{2}}^{(2)} \end{bmatrix} + \beta b_N b_N^t, \quad b_N = e_N^{(m)} + e_N^{(m+1)},$$

where the blocks $T_{\frac{N}{2}}^{(1)}$ and $T_{\frac{N}{2}}^{(2)}$ are $\frac{N}{2} \times \frac{N}{2}$ ST matrices and $\beta \equiv t_{m,m+1} \neq 0$ is the coupling term of these two blocks.

²Throughout the paper, vectors and matrices will be used with a subscript index denoting their dimension.

The *DC* algorithm [3][1][4] is based on the fact that in order to solve the eigenproblem of N -dimensional ST matrices, it is sufficient to solve this problem for $\frac{N}{2}$ -dimensional ST matrices. Specifically, if

$$(2.2) \quad \begin{aligned} T_{\frac{N}{2}}^{(1)} &= P_{\frac{N}{2}}^{(1)} \Lambda_{\frac{N}{2}}^{(1)} P_{\frac{N}{2}}^{(1)t}, & P_{\frac{N}{2}}^{(1)} P_{\frac{N}{2}}^{(1)t} &= I_{\frac{N}{2}} \\ T_{\frac{N}{2}}^{(2)} &= P_{\frac{N}{2}}^{(2)} \Lambda_{\frac{N}{2}}^{(2)} P_{\frac{N}{2}}^{(2)t}, & P_{\frac{N}{2}}^{(2)} P_{\frac{N}{2}}^{(2)t} &= I_{\frac{N}{2}}, \end{aligned}$$

are the spectral decompositions of the $\frac{N}{2} \times \frac{N}{2}$ ST matrices $T_{\frac{N}{2}}^{(1)}$ and $T_{\frac{N}{2}}^{(2)}$ respectively, then we can compute the spectral decomposition of the $N \times N$ ST matrix, T_N , by the following procedure:

I. First, we evaluate the unit N -vector z_N ,

$$(2.3a) \quad z_N = \frac{1}{\sqrt{2}} \begin{bmatrix} P_{\frac{N}{2}}^{(1)} & P_{\frac{N}{2}}^{(2)} \end{bmatrix}^t b_N, \quad b_N = \begin{bmatrix} 0 \\ \cdot \\ 1 \\ \dots \\ 1 \\ \cdot \\ 0 \end{bmatrix}$$

so that by (2.1), (2.2) and (2.3a), T_N is unitarily similar to the ROM matrix

$$\begin{aligned} T_N &= \begin{bmatrix} P_{\frac{N}{2}}^{(1)} \Lambda_{\frac{N}{2}}^{(1)} P_{\frac{N}{2}}^{(1)t} & \\ & P_{\frac{N}{2}}^{(2)} \Lambda_{\frac{N}{2}}^{(2)} P_{\frac{N}{2}}^{(2)t} \end{bmatrix} + \beta b_N b_N^t = \\ &= \begin{bmatrix} P_{\frac{N}{2}}^{(1)} & \\ & P_{\frac{N}{2}}^{(2)} \end{bmatrix} \left(\begin{bmatrix} \Lambda_{\frac{N}{2}}^{(1)} & \\ & \Lambda_{\frac{N}{2}}^{(2)} \end{bmatrix} + 2\beta z_N z_N^t \right) \begin{bmatrix} P_{\frac{N}{2}}^{(1)t} & \\ & P_{\frac{N}{2}}^{(2)t} \end{bmatrix} = \\ &= \begin{bmatrix} P_{\frac{N}{2}}^{(1)} & \\ & P_{\frac{N}{2}}^{(2)} \end{bmatrix} (D_N + 2\beta z_N z_N^t) \begin{bmatrix} P_{\frac{N}{2}}^{(1)} & \\ & P_{\frac{N}{2}}^{(2)} \end{bmatrix}^t, \quad D_N = \begin{bmatrix} \Lambda_{\frac{N}{2}}^{(1)} & \\ & \Lambda_{\frac{N}{2}}^{(2)} \end{bmatrix}. \end{aligned}$$

II. Second, we solve the updating problem – we find the spectral decomposition of the ROM matrix

$$(2.3b) \quad D_N + \sigma z_N z_N^t = Q_N \Lambda_N Q_N^t, \quad Q_N Q_N^t = I_N, \sigma = 2\beta.$$

III. Finally, we compute the unitary matrix

$$(2.3c) \quad P_N = \begin{bmatrix} P_{\frac{N}{2}}^{(1)} & \\ & P_{\frac{N}{2}}^{(2)} \end{bmatrix} Q_N ,$$

and obtain, by (2.3b) and (2.3c), the spectral decomposition of T_N as

$$T_N = \begin{bmatrix} P_{\frac{N}{2}}^{(1)} & \\ & P_{\frac{N}{2}}^{(2)} \end{bmatrix} Q_N \Lambda_N Q_N^t \begin{bmatrix} P_{\frac{N}{2}}^{(1)} & \\ & P_{\frac{N}{2}}^{(2)} \end{bmatrix}^t = P_N \Lambda_N P_N^t , \quad P_N P_N^t = I_N .$$

This process can be applied recursively: the N -dimensional eigenproblem of T_N is solved in terms of two independent $\frac{N}{2}$ -dimensional eigenproblems of $T_{\frac{N}{2}}^{(1)}$ and $T_{\frac{N}{2}}^{(2)}$, which in turn are solved in terms of four independent $\frac{N}{4}$ -dimensional eigenproblems of $T_{\frac{N}{4}}^{(1)}$, $T_{\frac{N}{4}}^{(2)}$, $T_{\frac{N}{4}}^{(3)}$, $T_{\frac{N}{4}}^{(4)}$, etc. Thus, the DC algorithm for an $N = 2^n$ -dimensional ST matrix, T_N , is organized as follows. After $n - 1$ splitting steps we are left with 2^{n-2} pairs of 2×2 ST matrices. In the first iteration they are used to construct, with the help of (2.3a)-(2.3c), the eigensystem of 2^{n-3} pairs of 4×4 ST matrices; in the second iteration, one constructs the eigensystem of 2^{n-4} pairs of 8×8 ST matrices, etc.; after $n - 2$ such iterations we end up with the eigensystems of the pair $T_{\frac{N}{2}}^{(1)}$, $T_{\frac{N}{2}}^{(2)}$, and the last $n - 1$ iteration solves the eigenproblem of T_N . A sequential implementation of a typical k -th iteration consists of 2^{n-k-1} times, evaluating the 2^{k+1} -dimensional unit vectors z in the first stage, (2.3a), solving 2^{k+1} -dimensional ROM eigenproblems in the second stage, (2.3b), and computing 2^{k+1} -dimensional products of unitary matrices in the third stage, (2.3c).

The total amount of work spent on the first two stages, (2.3a) and (2.3b), of all iterations, does not exceed $2\text{Const}.N^2$; the total work required for computing the eigenvectors in (2.3c) is $\frac{2}{3}N^3$. Thus, the total operations cost of the DC algorithm for finding the eigensystem (both the eigenvalues and eigenvectors) of an $N \times N$ ST matrix is $\frac{2}{3}N^3 + 2\text{Const}.N^2$.

If only the eigenvalues are required, then we can do better by saving the $O(N^3)$ operations required to compute the eigenvectors in the third stage (2.3c). Instead, the first stage of a typical k -th iteration, which requires 2^{n-k-1} different evaluations of 2^{k+1} -dimensional unit vectors of the form

$$z_{2^{k+1}} = \frac{1}{\sqrt{2}} \begin{bmatrix} P_{2^k}^{(1)} & \\ & P_{2^k}^{(2)} \end{bmatrix}^t b_{2^{k+1}} ,$$

can be efficiently implemented as follows: according to (2.3c), $P_{2^k}^{(1)}$ is represented by a successive product of

$$\begin{bmatrix} Q_{2^j}^{(1)} & & \\ & \ddots & \\ & & Q_{2^j}^{(2^{k-j})} \end{bmatrix} \quad j = k, k-1 \dots 1,$$

where $Q_{2^j}^{(\cdot)}$ were found by spectral decompositions of ROM matrices in previous iterations; similarly, $P_{2^k}^{(2)}$ is represented by a successive product of

$$\begin{bmatrix} Q_{2^j}^{(2^{k-j+1})} & & \\ & \ddots & \\ & & Q_{2^j}^{(2^{k-j+1})} \end{bmatrix}, \quad j = k, k-1 \dots 1.$$

Hence, we can evaluate each of the 2^{n-k-1} different vectors, $z_{2^{k+1}}$, as $z_{2^{k+1}} = z_{2^{k+1}}^{(k)}$, where

$$(2.4a) \quad z_{2^{k+1}}^{(j)} = \begin{bmatrix} Q_{2^j}^{(1)} & & \\ & \ddots & \\ & & Q_{2^j}^{(2^{k-j+1})} \end{bmatrix}^t z_{2^{k+1}}^{(j-1)}, \quad j = 1, 2 \dots k, \quad z_{2^{k+1}}^{(0)} \equiv \frac{1}{\sqrt{2}} b_{2^{k+1}},$$

at the expense of $\frac{1}{3}4^{k+1}$ operations. The total work spent on the first stages in all iterations is therefore $\leq \frac{2}{3}N^2$. This is complemented with the solution of 2^{n-k-1} different updating problems, see (2.3b)

$$(2.4b) \quad D_{2^{k+1}} + \sigma z_{2^{k+1}} z_{2^{k+1}}^t = Q_{2^{k+1}} \Lambda_{2^{k+1}} Q_{2^{k+1}}.$$

The total work spent on the second stage in all iterations amounts to $2\text{Const.}N^2$. Consequently, the total operations cost of the *DC* algorithm, (2.4a), (2.4b), for finding the eigenvalues of an $N \times N$ *ST* matrix is $(2\text{Const.} + \frac{2}{3})N^2$.

3. AN $O(N^2)$ METHOD FOR THE EIGENSYSTEM OF $N \times N$ ST MATRIX

Given an $N \times N$ ST matrix, T_N , we can compute its eigenvalues by the DC algorithm (2.4a), (2.4b) at the expense of no more than $O(N^2)$ operations.³ Thus, it remains to compute efficiently, i.e., with $O(N^2)$ operations, the eigenvectors of this matrix. To this end we may proceed as follows.

We seek for the unitary matrix P_N , $P_N P_N^t = I_N$, which diagonalizes T_N ,

$$(3.1) \quad T_N = P_N \Lambda_N P_N^t.$$

Let $p^{(i)} \equiv p_N^{(i)}$ denote the i -th row vector of P_N . Equating the i -th rows of

$$T_N P_N = P_N \Lambda_N$$

we obtain, in view of the reduced tridiagonal structure of T_N ,

$$(3.2) \quad t_{i,i-1} p_N^{(i-1)} + t_{i,i} p_N^{(i)} + t_{i,i+1} p_N^{(i+1)} = p_N^{(i)} \Lambda_N, \quad t_{i,i \pm 1} \neq 0.$$

Equation (3.2) is a linear three-term recurrence relation between the rows, $p_N^{(i)}$, of P_N , whose coefficients are determined by the entries of T_N . The input data required in order to solve these relations uniquely, consists of

1. The eigenvalues $\Lambda_N = \text{diag}(\lambda^{(1)}, \lambda^{(2)} \dots \lambda^{(N)})$ of T_N , which determine the terms $p_N^{(i)} \Lambda_N \equiv (\lambda^{(1)} p_{i1}, \lambda^{(2)} p_{i2} \dots \lambda^{(N)} p_{iN})$ on the right of (3.2). The eigenvalues are computed by the DC algorithm (2.4a), (2.4b) with $(2\text{Const.} + \frac{2}{3})N^2$ operations.
2. Two successive rows of P_N , which will serve as initial data for the recursive three-term relations (3.2). The proposed method hinges on the observation that two such rows emerge naturally from that part of the DC algorithm (2.4a), (2.4b) which computes the eigenvalues of T_N . Indeed, from the last $n - 1$ iteration of (2.4a) we have at our disposal the unit N -vector, z_N , which according to (2.3a) satisfies

³In fact, as observed by Cuppen [3], this number of operations can be substantially reduced - up to $O(N \log N)$ operations, in practical cases which employ sufficiently many deflations.

$$(3.3) \quad \begin{bmatrix} z_{\frac{N}{2}}^{(1)} \\ z_{\frac{N}{2}}^{(2)} \end{bmatrix} = \begin{bmatrix} P_{\frac{N}{2}}^{(1)} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & P_{\frac{N}{2}}^{(2)} \end{bmatrix}^t \begin{bmatrix} 0 \\ 1 \\ \cdots \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} z_{\frac{N}{2}}^{(1)} \\ z_{\frac{N}{2}}^{(2)} \end{bmatrix} \equiv \sqrt{2} \cdot z_N$$

Hence $z_{\frac{N}{2}}^{(1)}$ and $z_{\frac{N}{2}}^{(2)}$ are in fact the last and first column vectors of $P_{\frac{N}{2}}^{(1)t}$ and $P_{\frac{N}{2}}^{(2)t}$ respectively. Put differently, $(z_{\frac{N}{2}}^{(1)}, 0_{\frac{N}{2}})^t$ and $(0_{\frac{N}{2}}, z_{\frac{N}{2}}^{(2)})^t$ are rows number $m = \frac{N}{2}$ and $m + 1$ of $\begin{bmatrix} P_{\frac{N}{2}}^{(1)} \\ P_{\frac{N}{2}}^{(2)} \end{bmatrix}$. Consequently, equating the m and $m + 1$ rows of (2.3c) we obtain the two initial successive rows as

$$(3.4) \quad p_N^{(m)} = (z_{\frac{N}{2}}^{(1)}, 0_{\frac{N}{2}})^t Q_N,$$

$$p_N^{(m+1)} = (0_{\frac{N}{2}}, z_{\frac{N}{2}}^{(2)})^t Q_N;$$

the remaining rows of P_N are computed recursively by (3.2)

$$(3.5a) \quad p_N^{(i+1)} = \frac{1}{t_{i,i+1}} [p_N^{(i)}(\Lambda_N - t_{i,i}I_N) - t_{i,i-1}p_N^{(i-1)}], \quad i = m + 1 \dots N - 1,$$

$$(3.5b) \quad p_N^{(i-1)} = \frac{1}{t_{i,i-1}} [p_N^{(i)}(\Lambda_N - t_{i,i}I_N) - t_{i,i+1}p_N^{(i+1)}], \quad i = m, m - 1 \dots 2.$$

The operations cost of (3.4)-(3.5) does not exceed $3N^2$. Thus (2.4a),(2.4b) together with (3.4), (3.5) provide us with an $O(N^2)$ method for computing the whole eigensystem of an $N \times N$ ST matrices.

The error analysis of the proposed method depends on two ingredients:

1. The accuracy of the input data for (3.5a), (3.5b) – the eigenvalues $\Lambda_N = \text{diag}(\lambda^{(1)}, \lambda^{(2)} \dots \lambda^{(N)})$, and the two successive rows $p_N^{(m)}, p_N^{(m+1)}$ of P_N . This is determined by the stable behavior of the DC algorithm which was carefully analyzed in [3] [5]. Here we note that an accurate solution of the updating problems (2.4b) is essential for the stable behaviour of the DC algorithm. In Section 5 we discuss an improved version of the root finder [2] recommended by Cuppen [3], which accurately computes the eigenvalues Λ_N as the roots of the secular (characteristic) equation [6] associated with the ROM matrix $D_N + \sigma z_N z_N^t$ in (2.3b).

2. The second source of error is due to accumulation of rounding errors in the recurrence relations (3.5a), (3.5b). In order to examine this error accumulation, we rewrite (3.5) as a one-step iteration

$$(3.6a) \quad [p_N^{(i+1)}, p_N^{(i)}] = [p_N^{(i)}, p_N^{(i-1)}] \begin{bmatrix} \frac{1}{t_{i,i+1}}[\Lambda_N - t_{i,i}I_N] & I_N \\ -\frac{t_{i,i-1}}{t_{i,i+1}}I_N & 0_N \end{bmatrix}, \quad i = m+1 \dots N-1,$$

$$(3.6b) \quad [p_N^{(i-1)}, p_N^{(i)}] = [p_N^{(i)}, p_N^{(i+1)}] \begin{bmatrix} \frac{1}{t_{i,i-1}}[\Lambda_N - t_{i,i}I_N] & I_N \\ -\frac{t_{i,i+1}}{t_{i,i-1}}I_N & 0_N \end{bmatrix}, \quad i = m, m-1 \dots 2.$$

An indication to stability properties of (3.6a), (3.6b) is provided by the eigenvalues, $\kappa = \kappa_{ij}^{\pm}$ of the two $2N \times 2N$ matrices in the right-hand sides, i.e., for $i = m+1, m+2 \dots N-1$ we have

$$(3.7a) \quad t_{i,i+1}(\kappa_{ij}^{\pm})^2 - (\lambda_j - t_{i,i})\kappa_{ij}^{\pm} + t_{i,i-1} = 0, \quad j = 1, 2 \dots N$$

and for $i = m, m-1 \dots 2$ we have

$$(3.7b) \quad t_{i,i-1}(\kappa_{ij}^{\pm})^2 - (\lambda_j - t_{i,i})\kappa_{ij}^{\pm} + t_{i,i+1} = 0, \quad j = 1, 2 \dots N.$$

Hence the error in the i -th iteration of (3.5) is amplified by a factor of at least

$$g^{(i)} \equiv \max_{1 \leq j \leq N} (|\kappa_{ij}^+|, |\kappa_{ij}^-|).$$

Thus, the method is expected to be stable if

$$(3.8) \quad \prod_{i=2}^{N-1} g^{(i)} = \prod_{i=2}^{N-1} \max_{1 \leq j \leq N} (|\kappa_{ij}^+|, |\kappa_{ij}^-|) \leq \text{Const.}$$

As a canonical example for such stable behavior, let us consider ST matrices whose entries are 'slowly varying' along the diagonals, i.e., $t_{i,j} \sim t_{i+1,j+1}$. Now, if the superdiagonal entries are properly scaled so that also $t_{i,i+1} \sim t_{i,i-1}$, then by Gershgorin estimate we have for any $1 \leq j \leq N$,

$$|\lambda_j - t_{i,i}|^2 \lesssim (|t_{i,i-1}| + |t_{i,i+1}|)^2 \lesssim 4|t_{i,i-1}| \cdot |t_{i,i+1}|,$$

and hence the product of the characterisitic roots κ_{ij}^\pm is of order unity, for

$$|\kappa_{ij}^\pm|^2 = \left| \frac{(\lambda_j - t_{i,i}) \pm \sqrt{(\lambda_j - t_{i,i})^2 - 4t_{i,i+1}t_{i,i-1}}}{2t_{i,i\pm 1}} \right|^2 \lesssim \left| \frac{t_{i,i\mp 1}}{t_{i,i\pm 1}} \right| \sim 1.$$

If on the other hand (3.8) fails, we have an unstable error growth at the amount $\prod_{i=2}^{N-1} g^{(i)} \gg 1$, as confirmed by the numerical examples demonstrated in Section 6. Typically, such an instability shows up by the loss of orthogonality between the computed rows $p_N^{(i)}$ of P_N . Hence, one approach to solve the stability problem would be to use reorthogonalization, once the instability was detected by the loss of orthogonality, consult [3, Section 3]. An alternative approach to overcome the instability problem, which better suits the proposed method, is to *restart* the recurrence relations (3.5) at the current iteration with two new successive rows of P_N . How should we obtain such two successive rows to restart with? Consider for example the $N = 4m$ -dimensional problem. The iteration before the last of (2.4a) provides us with two $\frac{N}{2}$ -dimensional unit vectors, say $z_{\frac{N}{2}}$ and $w_{\frac{N}{2}}$, where

$$(3.9a) \quad \begin{bmatrix} z_{\frac{N}{4}}^{(1)} \\ z_{\frac{N}{4}}^{(2)} \end{bmatrix} = \begin{bmatrix} P_{\frac{N}{4}}^{(1)} & \vdots \\ \dots & \dots \\ & \vdots & P_{\frac{N}{4}}^{(2)} \end{bmatrix}^t \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} z_{\frac{N}{4}}^{(1)} \\ z_{\frac{N}{4}}^{(2)} \end{bmatrix} \equiv \sqrt{2} z_{\frac{N}{2}},$$

$$(3.9b) \quad \begin{bmatrix} w_{\frac{N}{4}}^{(1)} \\ w_{\frac{N}{4}}^{(2)} \end{bmatrix} = \begin{bmatrix} P_{\frac{N}{4}}^{(3)} & \vdots \\ \dots & \dots \\ & \vdots & P_{\frac{N}{4}}^{(4)} \end{bmatrix}^t \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} w_{\frac{N}{4}}^{(1)} \\ w_{\frac{N}{4}}^{(2)} \end{bmatrix} \equiv \sqrt{2} w_{\frac{N}{2}}.$$

As before, we obtain the m and $m+1$ rows of $P_{\frac{N}{2}}^{(1)}$ as

$$(3.10a) \quad \begin{aligned} p_{\frac{N}{2}}^{(1,m)} &= (z_{\frac{N}{4}}^{(1)}, 0_{\frac{N}{4}})^t Q_{\frac{N}{2}}^{(1)} \\ p_{\frac{N}{2}}^{(1,m+1)} &= (0_{\frac{N}{4}}, z_{\frac{N}{4}}^{(2)})^t Q_{\frac{N}{2}}^{(1)} \end{aligned}$$

and the m and $m+1$ rows of $P_{\frac{N}{2}}^{(2)}$ as

$$(3.10b) \quad \begin{aligned} p_{\frac{N}{2}}^{(2,m)} &= (w_{\frac{N}{4}}^{(1)}, 0_{\frac{N}{4}})^t Q_{\frac{N}{2}}^{(2)} \\ p_{\frac{N}{2}}^{(2,m+1)} &= (0_{\frac{N}{4}}, w_{\frac{N}{4}}^{(2)})^t Q_{\frac{N}{2}}^{(2)}. \end{aligned}$$

Consequently, we can compute with $O(N^2)$ operations rows number m , $m + 1$, $3m$, and $3m + 1$ of P_N , for by (2.3c) we have

$$\begin{aligned}
 p_N^{(m)} &= (p_{\frac{N}{2}}^{(1,m)}, 0_{\frac{N}{2}})^t Q_N \\
 p_N^{(m+1)} &= (p_{\frac{N}{2}}^{(1,m+1)}, 0_{\frac{N}{2}})^t Q_N \\
 p_N^{(3m)} &= (0_{\frac{N}{2}}, p_{\frac{N}{2}}^{(2,m)})^t Q_N \\
 p_N^{(3m+1)} &= (0_{\frac{N}{2}}, p_{\frac{N}{2}}^{(2,m+1)})^t Q_N
 \end{aligned}
 \tag{3.11}$$

In a similar manner one can restart the recurrence relations (3.5) at any desired iteration.

4. THE EIGENVECTORS OF T_N - ONE AT THE TIME

In the previous section we discussed an $O(N^2)$ method for computing the whole eigen-system - eigenvalues and eigenvectors of an $N \times N$ ST matrix. In several applications one is interested in only a few of the eigenvectors of T_N . We now present a variant of this method which enables one to compute each one of the desired eigenvectors with $O(N)$ operations.

As before, we first prepare, with the help of the DC algorithm (2.4a), (2.4b), (3.4), the eigenvalues $\{\lambda^{(j)}\}_{j=1}^N$ and the two middle successive rows, $p_N^{(m)}$ and $p_N^{(m+1)}$ of P_N . This can be done at the expense of $O(N^2)$ operations, and in many practical cases with even less. Equipped with this we can compute the eigenvector $x_N = (x^{(1)}, x^{(2)} \dots x^{(N)})$ corresponding to the eigenvalue, say, $\lambda^{(j)}$

$$(4.1) \quad T_N x_N = \lambda^{(j)} x_N .$$

Equation (4.1) gives us the linear three term recurrence relations between the components of x

$$(4.2) \quad t_{i,i-1}x^{(i-1)} + t_{i,i}x^{(i)} + t_{i,i+1}x^{(i+1)} = \lambda^{(j)}x^{(i)} .$$

Since x_N coincides with the j -th column of P_N , we have its two middle entries $x^{(m)}$ and $x^{(m+1)}$ from the j -th entries of $p_N^{(m)}$ and $p_N^{(m+1)}$. The rest of the entries are computed recursively with $3N$ operations by

$$(4.3a) \quad x^{(i+1)} = \frac{1}{t_{i,i+1}} \left[(\lambda_j - t_{i,i})x^{(i)} - t_{i,i-1}x^{(i-1)} \right], \quad i = m+1 \dots N-1 ,$$

$$(4.3b) \quad x^{(i-1)} = \frac{1}{t_{i,i-1}} \left[(\lambda_j - t_{i,i})x^{(i)} - t_{i,i+1}x^{(i+1)} \right], \quad i = m, m-1 \dots 2 .$$

The computation is stable or unstable depending on whether

$$(4.4) \quad \prod_{i=2}^{N-1} \max(|\kappa_{ij}^+|, |\kappa_{ij}^-|)$$

is either bounded or $\gg 1$.

5. SOLUTION OF THE UPDATING PROBLEM

In this section we follow [1] and [3] in a discussion on the promised $O(N^2)$ method for solving the updating problem (2.3b), i.e., computing the eigensystem of $D_N + \sigma z_N z_N^t$. Without loss of generality we may assume that $\sigma > 0$ and that the problem has been deflated, so that the components of $z_N = (z^{(1)} \dots z^{(N)})^t$ as well as the difference between any two diagonal entries of $D_N = \text{diag}(d_{11} < d_{22} < \dots < d_{NN})$ are different from zero (in practice we take a neighbourhood of zero with a preassigned tolerance, say ϵ), consult [1], [3], and [4, Section 4]. In such case, it follows that the eigenvalues of the updating problem $\lambda^{(i)}, i = 1, 2, \dots, N$, strictly interlace with those of D_N [1, Theorem 1], [3, Theorem 2.1]

$$(5.1) \quad d_{11} < \lambda^{(1)} < d_{22} < \lambda^{(2)} < \dots < \lambda^{(N)} < d_{NN} + \sigma \equiv d_{N+1, N+1}$$

With this in mind we now turn to compute the required eigenvalues, $\lambda = \lambda^{(i)}$, as the roots of the so called secular equation [6]

$$(5.2) \quad f(\lambda) \equiv 1 + \sigma \sum_{j=1}^N \frac{(z^{(j)})^2}{d_{jj} - \lambda} = 0.$$

The function $f(\lambda)$ is the rational representation of the characteristic polynomial associated with $D_N + \sigma z_N z_N^t$, and the interlacing property ensures that f has N simple roots, $\lambda^{(i)}$, lying in the open intervals $(d_{ii}, d_{i+1, i+1}), i = 1, 2, \dots, N$. We shall mention two zero-finders which have been advocated to find these roots:

1. The zero-finder proposed by Bunch et al. [1] which is based on rational interpolation, employs the values of $f(\lambda)$ and its first derivative, $f'(\lambda)$. The advantage of this zero-finder (which will be referred to as 'zeroinder') is that it produces a *monotonic* sequence of approximations in $(d_{ii}, d_{i+1, i+1})$ which converges quadratically to $\lambda^{(i)}$. However, it is very sensitive near the ends of the intervals, $(d_{ii}, d_{i+1, i+1})$, where the derivative involved, $f'(\lambda)$, become singular.
2. Cuppen [3] advocated the 'zeroinderat' zero-finder of Bus and Dekker [2] which is based on rational interpolation of three f -values in the interval $(d_{ii}, d_{i+1, i+1})$. This

algorithm is more robust than the 'zeroinder', for it does not involve $f'(\lambda)$; consequently, it avoids the previous difficulty of singular derivatives near d_{ii} and moreover, it saves half the operations per iteration. Yet, the current 'zeroinrat' algorithm lacks the monotonicity property we had before, and therefore, it requires safeguarding to ensure that we remain within the desired interval (this decreases the convergence rate to ~ 1.839).

Assuming that either one of these zero-finders requires no more than a constant number of iterations to compute (with some preassigned tolerable accuracy) each root of (5.2), then the required eigenvalues $\lambda^{(i)}, i = 1, 2 \dots N$, are obtained by $O(N^2)$ operations. Equipped with these eigenvalues we now may use the Sherman-Morrison formula to compute the associated normalized eigenvectors, $q_N^{(i)}$, which form the column vectors of Q_N in (2.3b), as [1, Section 4]

$$(5.3) \quad q_N^{(i)} = \frac{(D_N - \lambda^{(i)} I_N)^{-1} z_N}{\|(D_N - \lambda^{(i)} I_N)^{-1} z_N\|}, \quad i = 1 \dots N,$$

and the total operations cost does not exceed $O(N^2)$, as asserted.

To enhance the stability properties of the whole *DC* algorithm, the updating problem should be solved with maximum accuracy. To achieve this, we now present an efficient implementation for the solution of this problem, based on the ingredients described above.

As a first step we reformulate (5.2) in a manner suggested in [1]. By the interlacing property (5.1) we have

$$(5.4) \quad \lambda^{(i)} = d_{ii} + \sigma \mu^{(i)}, \quad 0 < \mu^{(i)} < 1, \quad \sum_{i=1}^N \mu^{(i)} = 1.$$

For $i = 1, 2, \dots, N$ we make the change of variables, $\lambda = d_{ii} + \sigma \mu$, so that instead of $f \equiv f(\lambda)$, we now obtain N different rational functions, $W_i(\mu)$,

$$(5.5) \quad W_i(\mu) = 1 + \sum_{j=1}^N \frac{(z^{(j)})^2}{\delta_{ji} - \mu}, \quad \delta_{ji} \equiv \frac{d_{jj} - d_{ii}}{\sigma}, \quad i = 1, 2 \dots N,$$

each of which has a simple root, $\mu = \mu^{(i)}$, in the open interval $(\delta_{ii} \equiv 0, \delta_{i+1,i})$. Computing the root of $W_i(\mu)$ in this interval – rather than the root of $f(\lambda)$ in the $(d_{ii}, d_{i+1,i+1})$ interval

– has the advantage that $W'_i(\mu)$ is *uniformly* bounded from below (by 1) rather than having $f'(\lambda) \geq \frac{1}{\sigma}$, as in [3, Theorem 3.1]. The computation of the desired root proceeds by carefully monitoring a mixture of the two zero-finders mentioned above. Namely, the ‘zeroinder’ algorithm will be used when we are well inside the interval of interest $(0, \delta_{i+1,i})$, while we switch to the ‘zeroinrat’ algorithm when we approach either end of this interval. To decide upon the switching policy we first quote from [5] (see [4]).

LEMMA 5.1. *Assume that the deflation test $|z^{(i)}| > \epsilon$ is satisfied. Then either we have*

$$(5.6a) \quad \frac{\epsilon^2}{\sigma^2(2 + \delta_{i+1,i})} \delta_{i+1,i} \leq \mu^{(i)} \leq \frac{1}{2} \delta_{i+1,i}$$

or alternatively

$$(5.6b) \quad \frac{1}{2} \delta_{i+1,i} \leq \mu^{(i)} \leq \left(1 - \frac{\epsilon^2}{\sigma^2(2 + \delta_{i+1,i})}\right) \delta_{i+1,i}$$

The bounds on the left and right-hand-sides of (5.6) yield a closed subinterval $[L^{(i)}, H^{(i)}]$ which encloses $\mu^{(i)}$. (Experiments have shown that these bounds may actually be achieved).

A more practical indication to the location of $\mu^{(i)}$ is obtained from the following considerations. The rational function

$$(5.7a) \quad V_i(\mu) = \text{Const}_i + \frac{(z^{(i)})^2}{-\mu} + \frac{(z^{(i+1)})^2}{\delta_{i+1,i} - \mu}, \quad \text{Const}_i \equiv \sum_{j \neq i, i+1} \frac{(z^{(j)})^2}{\delta_{ji} - \delta_{i+1,i}}$$

has a simple root, $\ell^{(i)}$, in the interval $(0, \delta_{i+1,i})$. Since $V_i(\mu)$ dominates $W_i(\mu)$ in that interval and they are both monotonically increasing, we can use this root (which is found by solving a simple quadratic equation) as a lower bound for $\mu^{(i)}$. Similarly, the function

$$(5.7b) \quad U_i(\mu) = \text{Const}_i + \frac{(z^{(i)})^2}{-\mu} + \frac{(z^{(i+1)})^2}{\delta_{i+1,i} - \mu}, \quad \text{Const}_i \equiv \sum_{j \neq i, i+1} \frac{(z^{(j)})^2}{\delta_{ji}}$$

has a simple root, $h^{(i)}$, in the interval $(0, \delta_{i+1,i})$ which may serve as an upper bound for $\mu^{(i)}$.

Returning to our problem of finding the roots of $W_i(\mu)$ in (5.5), we use the ‘zeroinder’ algorithm when inside the $(0, \delta_{i+1,i})$ interval. This requires us to compute $W'_i(\mu)$, and

Lemma 5.1 indicates that as we approach either end of the interval, the computation of $W'_i(\mu)$ involves factors of ϵ^{-4} which will lead to an underflow problem. To avoid this situation we use a switching policy, which in each step tests if either one of the following inequalities is satisfied

$$(5.8) \quad \ell^{(i)} \leq L^{(i)} , \quad h^{(i)} \geq H^{(i)} , \quad h^{(i)} \leq \ell^{(i)}$$

as an indication that we are in the neighborhood of the singular ends, in which case we use the 'zeroinrat' algorithm instead. This 'switching' policy enabled us to achieve with the usual 64-bit arithmetic, more than satisfactory results that otherwise would have required the less attractive extended precision arithmetic.

Concerning the computation of the eigenvectors in (5.3), we note that it is possible to have severe round-off when $\lambda^{(i)}$ is close to d_{ii} or $d_{i+1,i+1}$ [3, Section 2]. The reformulation of the eigenvalue problem in (5.5) enables one to avoid half of these round-off problems, namely when $\lambda^{(i)}$ is close to d_{ii} . Indeed, the *normalized eigenvectors*, $q_N^{(i)}$, are now given by

$$(5.9) \quad q_N^{(i)} = \frac{[D_N^{(i)}]^{-1} z_N}{\|[D_N^{(i)}]^{-1} z_N\|} , \quad D_N^{(i)} = \text{diag}(\delta_{1i} \dots \delta_{Ni}) - \mu^{(i)} I .$$

Using (5.9) instead of (5.3) avoids cancellation which arises when $\lambda^{(i)}$ is too close to d_{ii} , i.e., when $\mu^{(i)}$ is too close to zero, for $\delta_{ii} \equiv 0$ in this case. We are still left with the other half of the cancellation problem when $\lambda^{(i)}$ is too close to $d_{i+1,i+1}$. If this is indeed the case (as we can foresee by computing the practical bounds for $\mu^{(i)}$ from (5.7a), (5.7b)), then we propose to perform yet another reformulation of our eigenvalue problem, using

$$(5.10) \quad \lambda^{(i)} = d_{i+1,i+1} - \sigma \eta^{(i)}$$

instead of (5.4). In this case the role of the rational functions $W_i(\mu)$ in (5.5) is played by

$$(5.11) \quad \overline{W}_i(\eta) = 1 + \sum_{j=1}^N \frac{(z^{(j)})^2}{\delta_{j,i+1} - \eta} , \quad \delta_{j,i+1} = \frac{d_{jj} - d_{i+1,i+1}}{\sigma} , \quad i = 1, 2 \dots n$$

each of which has a simple root $\eta = \eta^{(i)}$ in the open interval $(0, -\delta_{i,i+1})$. The corresponding normalized eigenvectors are given by

$$(5.12) \quad q_N^{(i)} = \frac{[\overline{D}_N^{(i)}]^{-1} z_N}{\|[\overline{D}_N^{(i)}]^{-1} z_N\|}, \quad \overline{D}_N^{(i)} = \text{diag}(\delta_{1,i+1} \dots \delta_{N,i+1}) + \eta^{(i)} I_N,$$

and cancellation which arises when $\lambda^{(i)}$ is too close to $d_{i+1,i+1}$, i.e., when $\eta^{(i)}$ is too close to zero is avoided for $\delta_{i+1,i+1} \equiv 0$.

6. NUMERICAL EXPERIMENTS

The main object of our experiments was a comparison between the standard $O(N^3)$ DC algorithm for computing the eigensystems of $N \times N$ ST matrices, and the proposed $O(N^2)$ method in (2.4a), (2.4b) and (3.4), which makes use of the three-term recurrence relations (3.5a), (3.5b). The input data for these relations – the eigenvalues $\lambda^{(i)}$ and the two initial successive row vectors $p_N^{(m)}, p_N^{(m+1)}$, were supplied with maximum accuracy, with the help of the updating solver described in Section 5 which avoids extended precision. Indeed all our calculations, including the pathologically ill-conditioned W_{+N} -Wilkinson's matrices, were carried out with a 64-bit arithmetic.

The first set of results includes 'well-behaved' matrices taken from [7, (7.4)-(7.9)]. The entries along the diagonals of these matrices are 'slowly varying' and their eigenvalues are equally distributed. The stability analysis in Section 3 indicates bounded amplification factors in these cases, and the numerical results confirmed the expected stable behavior of our method. Table 1 summarizes the results for the prototype ST matrix of this group where $T_{ij} = 2 - |i - j|$.

	Standard DC Algorithm		The Proposed Method	
N	$\ TP - P\Lambda\ _\infty$	$\ P^t P - I\ _\infty$	$\ TP - P\Lambda\ _\infty$	$\ P^t P - I\ _\infty$
101	2.5E-15	6.2E-16	9.5E-15	7.2E-15
201	2.6E-15	2.5E-15	2.2E-14	1.5E-14
301	3.0E-15	2.8E-15	2.9E-14	8.8E-14
401	4.0E-15	6.9E-15	2.5E-13	1.2E-13

Table 1: Results for $T[1, 2, 1]$ matrix of order N :

Since the rows of P were constructed by equating to zero rows $2, 3 \dots N - 1$ of $TP - P\Lambda$, the quantities on the left columns, $\|TP - P\Lambda\|_\infty$, stand for

$$\max(\|t_{1,1}p^{(1)} + t_{1,2}p^{(2)} - p^{(1)}\Lambda\|, \|t_{N,N-1}p^{(N-1)} + t_{N,N}p^{(N)} - p^{(N)}\Lambda\|) .$$

They may serve us as a quantitative indication of the accumulation of rounding errors in the three-term recursion relations (3.5), which is responsible for the loss of (no more than) two orders of magnitude relative to the standard algorithm. The advantage of the proposed method lies upon the fact that the results on the right columns are obtained by saving order of magnitude in execution time relative to the results on the left columns.

Next, we turn to the second group of matrices which consist of Wilkinson's matrices, W_{+N} . The superdiagonals of these matrices are properly scaled to begin with – they all equal one; the entries along the main diagonal, however, $\text{diag}(\frac{N-1}{2}, \frac{N-3}{2}, \dots, 1, 0, 1, \dots, \frac{N-1}{2})$ are far from being 'slowly varying'. This leads to amplification factors of the recurrence relations (3.5) of order $\sim \frac{N-1}{2}!$, which indicates loss of all (64-bit precision) significant figures in computing the eigenproblem of W_{+N} of order $N \gtrsim 40$. Moreover, the largest eigenvalues of W_{+N} are clustered in pairs, which may be inseparable up to the 14th decimal digit. This then leads to additional inaccuracies in the updating solution (while seeking two extremely close roots of the secular equation) as well as in the deflation process. As a result, the initial input data for the recurrence relation will also suffer from loss of accuracy. These arguments are well reflected in the following table:

	Standard DC Algorithm		The Proposed Method	
N	$\ TP - P\Lambda\ _\infty$	$\ P^t P - I\ _\infty$	$\ TP - P\Lambda\ _\infty$	$\ P^t P - I\ _\infty$
21	4.5E-16	2.5E-16	1.2E-12	9.8E-10
41	1.3E-15	9.4E-16	3.7E-8	7.4E-7
47	2.0E-15	9.1E-16	5.3E-3	1.0E-3
49	2.0E-15	9.8E-16	0.12	0.23

Table 2: Results for the W_{+N} matrices.

An attempt to improve the results of our method was made, in order to be competitive with the standard algorithm which gave excellent results for W_{+N} up to order $N = 201$. To this end we have appended our method with the *restarting* procedure described in Section 3. Thus, by computing the row vectors (here $m = \frac{N+1}{4}$) $p_N^{(m)}$, $p_N^{(m+1)}$, $p_N^{(3m)}$ and $p_N^{(3m+1)}$ as additional input data to restart the three term recurrence relations we were able to get decent results for the W_{+N} -matrices up to order $N \sim 200$. A finite number of such restarting procedures would enable us to deal with even larger W_{+N} -matrices, still within the $O(N^2)$ operations limit.

Finally, the last group of matrices that were tested consists of randomly generated entries in $[-1, 1]$. The results obtained are summarized in Table 3.

	Standard DC Algorithm		The Proposed Method	
N	$\ TP - P\Lambda\ _\infty$	$\ P^tP - I\ _\infty$	$\ TP - P\Lambda\ _\infty$	$\ P^tP - I\ _\infty$
100	8.4E-15	9.8E-16	9.5E-15	7.6E-15
200	5.9E-15	3.4E-15	6.2E-9	9.8E-8
300	6.3E-15	5.6E-15	4.2E-4	3.1E-2
400	7.2E-15	6.8E-15	O(1)	O(1)

Table 3: Results for random matrices of order N .

We observe that excellent results are obtained by our method for such randomly generated matrices of order up to $N \sim 100$. If additional restarting procedures are employed every 100 – 200 iterations, it would enable us to achieve highly accurate results for matrices of almost any practical size.

In summary, we conclude that the proposed method for solving the eigenproblem of ST matrices, provides a competitive alternative to the standard eigensolvers for a *wide* class of such matrices; by sacrificing a *few* orders of accuracy, the method enables one to save *order of magnitude* in the total execution time. This conclusion was confirmed by further extensive numerical experiments reported in [5].

References

- [1] Bunch, H.J.R., Nielsen, C.P., Sorensen, D.C.: Rank one modification of the symmetric eigenproblem, *Numer. Math.* 31, 31-48 (1978).
- [2] Bus, J.C., Dekker, T.J.: Two efficient algorithms with guaranteed convergence for finding a zero of a function, *TOMS* 1, 330-345 (1975).
- [3] Cuppen, J.J.M.: A Divide and conquer method for the symmetric tridiagonal eigenproblem, *Numer. Math.*, 36, 177-195 (1981).
- [4] Dongarra, J.J., Sorensen, D.C.: A fully parallel algorithm for the symmetric eigenvalue problem, *SIAM J. Sci. Stat. Comput.* 8, 139-154 (1987).
- [5] Gill, D., Divide and Conquer – A parallel algorithm for computing the spectral decomposition of symmetric matrices, M.Sc. Thesis, Tel-Aviv University, 1987.
- [6] Golub, G.H.: Some modified matrix eigenvalue problems, *SIAM Rev.* 15, 318-334 (1973).
- [7] Gregory, R.T., Karney, D.L.: A collection of matrices for testing computational algorithms, New York: John Wiley 1969.
- [8] Parlett, B.N.: The symmetric eigenvalue problem, Prentice Hall, Inc., Englewood Cliffs, N.J., 1980.
- [9] Peters, G. and Wilkinson, J.H.: Eigenvalues of $Ax = \lambda Bx$ with bound symmetric A and B , *Comput. J.* 12, 398-404 (1969).
- [10] Wilkinson, J.H.: The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.



Report Documentation Page

1. Report No. NASA CR-181647 ICASE Report No. 88-19		2. Government Accession No. AD-A192 761		3. Recipient's Catalog No.	
4. Title and Subtitle AN $O(N^2)$ METHOD FOR COMPUTING THE EIGENSYSTEM OF $N \times N$ SYMMETRIC TRIDIAGONAL MATRICES BY THE DIVIDE AND CONQUER APPROACH				5. Report Date March 1988	
				6. Performing Organization Code	
7. Author(s) Doron Gill and Eitan Tadmor				8. Performing Organization Report No. 88-19	
				10. Work Unit No. 505-90-21-01	
9. Performing Organization Name and Address Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665-5225				11. Contract or Grant No. NAS1-18107	
				13. Type of Report and Period Covered Contractor Report	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225				14. Sponsoring Agency Code	
15. Supplementary Notes Langley Technical Monitor: Richard W. Barnwell Submitted to SIAM J. Sci. Statist. Comput. Final Report <i>N-cubed</i>					
16. Abstract <i>is proposed</i> We propose an efficient method, to solve the eigenproblem of $N \times N$ Symmetric Tridiagonal (ST) matrices. Unlike the standard eigensolvers which necessitate $O(N^3)$ operations to compute the eigenvectors of such ST matrices, the proposed method computes both the eigenvalues and eigenvectors with only $O(N^2)$ operations. The method is based on <u>serial</u> implementation of the recently introduced Divide and Conquer (DC) algorithm [3],[1],[4]. It exploits the fact that by $O(N^2)$ of DC operations, one can compute the eigenvalues of $N \times N$ ST matrix and a finite number of pairs of successive rows of its eigenvector matrix. The rest of the eigenvectors--all of them or one at the time, are computed by linear three-term recurrence relations. We conclude with numerical examples, which demonstrate the superiority of the proposed method by saving an <u>order</u> of magnitude in execution time at the expense of sacrificing a <u>few</u> orders of accuracy. <i>key: etc</i>					
17. Key Words (Suggested by Author(s)) symmetric eigenvalue problem; divide and conquer; updating problem			18. Distribution Statement 64 - Numerical Analysis Unclassified - unlimited		
19. Security Classif. (of this report) Unclassified		20. Security Classif. (of this page) Unclassified		21. No. of pages 24	22. Price A02

END

DATE

FILMED

6-88

DTIC